REMARKS

Claims 1-47 remain in the application for consideration. In view of the following remarks, Applicant respectfully requests the application be forwarded to issuance.

The Specification

The disclosure is objected to as containing an embedded hyperlink. Applicant has amended the specification to remove the hyperlink thereby overcoming the objection. Applicant has reviewed the specification and believes that all of the browser-executable code has been removed.

§103 Rejections

Claims 1-11, 13, 14, 16-19, 31-35, and 38 stand rejected under 35 U.S.C. §103(a) as being unpatentable over U.S. Patent No. 6,012,098 to Bayeh et al. (hereinafter "Bayeh") in view of U.S. Patent No. 6,249,844 to Schloss et al. (hereinafter "Schloss"). Applicant disagrees with the Office's rejections of these claims and, in view of the discussion below, respectfully traverses the rejections.

Claims 20-26, 30, 41-42, and 44-47 stand rejected under §103(a) over Bayeh and Schloss in view of a document to Whitehead entitled "WEBDAV: IETF Standard for Collaborative Authoring on the Web" (hereinafter "Whitehead").

Claims 12, 15, 36, 39, and 40 stand rejected under §103(a) over Bayeh and Schloss in view of U.S. Patent No. 6,366,947 to Kavner.

Claims 27-29 and 43 stand rejected under §103(a) over Bayeh and Schloss in view of Whitehead and Kavner.

Claim 37 stands rejected under §103(a) over Bayeh.

Before undertaking a discussion of the Office's application of Bayeh and Schloss, as well as the other references, to the claims, the following discussion of Bayeh and Schloss is provided to aid the Office in appreciating some of the differences between the subject matter described in the cited references and the various claimed embodiments.

The Bayeh Reference

Bayeh is directed to a method and system that utilizes servlet pairing for isolation of the retrieval and rendering of data. In accordance with its disclosure, data retrieval logic is isolated to a data servlet, and presentation formatting is isolated to a rendering servlet. Servlet chaining is used to send the output of the data servlet to the rendering servlet. The data servlet formats its output data stream for transfer to a downstream servlet. This data stream is formatted using a language such as the Extensible Markup Language (XML), according to a specific Document Type Definition (DTD). The rendering servlet parses this XML data stream, using a style sheet that may be written using the Extensible Style Language (XSL), and creates a HyperText Markup Language (HTML) data stream as output.

Bayeh describes the processing that takes place on the server side as follows:

> At Step 230, the server which received the client's request routes it to the proper data servlet. ***

Steps 240 through 270 are implemented by a data servlet according to the present invention. These steps represent the logic associated with data retrieval, as well as minimal formatting of the data for transfer to a rendering servlet: the formatting is not a presentation format.

At Step 240, the data servlet processes the client request. The request will typically require retrieving data from some database available to the data servlet. The data servlet will format a database query request, using an appropriate query language that will depend on the type of database on which the relevant data is stored. ***

Step 250 is included to indicate that, optionally, additional processing may be performed on the data received in response to the database query. ***

At Step 260, the data servlet formats the database information as an XML data stream. As previously discussed, a DTD is used in this formatting step. The DTD specifies how specific predefined "tags" are to be inserted into the XML data stream. A tag is a keyword that identifies what the data is which is associated with the tag, and is typically composed of a character string enclosed in special characters. "Special characters" means characters other than letters and numbers, which are defined and reserved for use with tags. Special characters are used so that a parser processing the data stream will recognize that this a tag. A tag is normally inserted preceding its associated data: a corresponding tag may also be inserted following the data, to clearly identify where that data ends. ***

When the entire XML data stream required for the database results has been formatted by the data servlet, that data stream is sent on to the next servlet in the chain at Step 270. In the preferred embodiment, the next servlet is the rendering servlet.

Steps 280 through 320 are implemented by a rendering servlet according to the present invention. These steps represent the logic associated with data presentation formatting.

Step 280 receives the XML data stream created by, and sent by, the data servlet. Techniques for receiving a data stream from a chained servlet are well known in the art, and Step 280 is shown for completeness of depicting the function of the rendering servlet.

According to the preferred embodiment, the rendering servlet ***must*** parse the XML data stream, and reformat it into HTML. ***This is necessary*** because browsers, by convention, expect to receive data that has been formatted with HTML. As discussed previously, this parsing process requires two types of data input: the XML data stream, and style sheet information. In the preferred embodiment, an XSL style sheet is used. Techniques for writing parsers are well known in the art, and will not be described in detail herein.

Step 290 indicates that the rendering servlet locates the XSL style sheet. Typically, the style sheet will be stored as a file on a medium, such as a disk drive, accessible to the rendering servlet. Alternatively, the style sheet might be incorporated directly into the code of the rendering servlet; however, because incorporating the information directly into the code makes revising the style sheet more difficult, this alternative is less desirable than isolating the style sheet as a separate file.

At Step 300, the rendering servlet parses the XML data stream, using the input sources described above. A parser reads a data stream, looking for predefined strings of characters that the parser recognizes, and which indicate to the parser what type of data is represented. In the preferred embodiment, the DTD used in the data servlet specified predefined strings that were used as tags, as described above, and inserted into the XML data stream by the data servlet. The parser looks for these tags as it reads the XML data stream. When a recognized tag is found, the parser knows what type of XML document element appears in the data stream between this tag and its corresponding ending tag (or following this tag, if ending tags are not required). As the parser in the rendering servlet determines what each document element is, it creates a new data stream, formatted using HTML. The parser may also insert presentation style attributes into the HTML data stream, where the appropriate attribute to use is determined according to the type of document element the parser is currently processing. Creation of the HTML data stream is represented in FIG. 5 as Step 310, although one skilled in the art will recognize that the functions of Steps 300 and 310 are intermingled. That is, as the parser processes portions of the input XML data stream, it creates the corresponding HTML data stream, then processes another portion of the input data stream, creates more of the output data stream, etc., until the entire input data stream has been processed.

When the reformatted data stream is complete, Step 320 sends that HTML data stream back to the client's browser, to be processed by the browser for presentation to the user.

The Schloss Reference

Schloss discloses methods and systems for identifying and creating persistent object fragments from a named object. Schloss is perhaps best understood in the context of its "Background" section which identifies some of the problems that Schloss addresses.

Specifically, as noted by Schloss, using formal descriptors, such as a markup language like XML, to describe a digital document provides tremendous flexibility. The markup language description can provide rich information on the document structure and the final document to be generated. In fact, XML is a language that allows users to define their own language. For example, chemists can define a chemical markup language to describe a molecular structure. Mathematicians or scientists can define a math markup language to describe complex mathematical formulas. The interpretation of the markup language description and generation of the object can thus be complex. It is desirable to avoid regeneration of the same description repeatedly. Since Web pages, objects or documents on a common subject, or from the same company or authors often have parts in common, there is a need to go beyond recognizing just the repeated references to named entities (i.e., subject already has a name, e.g., URL) to subparts of named entities.

Schloss instructs that proxy or Web servers and client browsers today do not interpret the markup language to decompose a document or object into components, provide persistent identities and tracking mechanisms to facilitate caching and recognition of repeated occurrences of components of a named object. They mainly provide caching or processing service for named objects as a whole. For example, as mentioned previously, in HTML the text documents and images (which are separated out from the text documents by the authors) are all named objects and hence cacheable entities.

Another problem, as Schloss instructs, is that if a document includes dynamic content, caching is not meaningful as the next reference to the same document URL can result in a different version of the document. Thus a document is not cached even if only a small fraction of its content is dynamic. This is an issue for HTML documents today and is expected to become more severe for XML documents, which are more flexible and make it easier to incorporate various types of dynamic information, such as data from a database.

Thus, Schloss submits that a need remains for a system and method for identifying and creating one or more persistent object fragments from a named object, for example to facilitate caching.

As described by Schloss, its methods and systems are directed to identifying and creating persistent object fragments from a named object. In accordance with Schloss's described techniques, methods and apparatus can dynamically parse a digital content description of a named digital object, and create and maintain fragment identities to facilitate caching.

In accordance with Schloss's methods and systems, various features can parse/analyze the object description, identify object fragments and create persistent object fragment identities, and revise the object description by replacing each object fragment with its newly created persistent identity and send the revised object description to a requesting node. Depending upon the properties of a fragment, this can either enable the fragment to be cacheable (which can be at the content/proxy server and the client device in the Web environment), or make the revised object description cacheable at the server and client device.

Schloss's approach can be most easily appreciated from the example that Schloss sets out and describes in connection with Figs. 3 and 4. Specifically, FIG. 3 depicts an example of a document with 3 SEGMENTS. The first segment (330.sub.1) begins with a start-tag, <cml: molecule>, and ends with an end-tag, </cml: molecule>, and the second segment begins with a start-tag, <m: order>, and ends with an end-tag, </m: order>. The second segment (330.sub.2) includes a third segment (330.sub.3) nested within it. The third segment begins with a start-tag, <db: price>, and finishes with an end-tag, </db: price>. In this example, Schloss asks the reader to assume the semantics of the three segments as follows. Assume the first segment provides an image of a molecule structure of a chemical compound. Assume also the second segment contains a formula to generate an order table showing the price at different quantities. Assume further, the third segment retrieves the price information from the product database. Hence it is a segment with dynamic information.

FIG. 4 depicts an example of a modified Web document after the persistent fragments have been recognized and extracted. Here, as noted by Schloss, it is assumed that generating the molecular structure of the chemical compound in the first segment (330.sub.1) is quite complex, whereas the computation of the order table is straightforward. Hence, only the first (330.sub.1 ') and the third segments (330.sub.3 ') are recognized as persistent fragments with the identities, "125.1" and "28.3", respectively. In accordance with Schloss's example, each of the persistent fragments is replaced with an "include" statement referring to the name of the fragment, e.g. <include HREF="125.1">, indicating the reference to the fragment "125.1," and followed by a <include> statement.

Schloss further describes, in Fig. 5, an example of a fragment description table for tracking the object fragment identity and its description.

The remainder of Schloss is dedicated to describing how the particular fragmentation process is performed.

Applicant's Disclosure

Applicant's disclosure describes various methods and systems for generating and sending XML documents and, in particular, generating and sending an XML response to an XML client request.

In various described embodiments and not necessarily all of the described and claimed embodiments, an XML document is prepared and sent to a client only a portion at a time. XML document portions are generated and sent until an entire XML document is sent to the client. In a specific implementation an instance of which is claimed in one claim set, an XML response generator is provided and responds to a client request without having to first build and save a hierarchical tree structure in memory that represents the response. The response generator includes one or more request method objects. There is, in this embodiment, one request method object for each particular type of client request that might be received. Each request method object knows and gathers the data that is needed to respond to its particular associated client request. In addition, the request method object knows the order in which the information must be provided.

In accordance with one embodiment, the request method object calls an emitter object with the data that is gathered by the request method object. The calls are made in a particular order and ensure that the hierarchical nature of the response that is being built is preserved. The emitter object translates the data that it receives into response portions that are in proper XML syntactic form.

In accordance with one embodiment, a body object is provided to manage a buffer. The emitter object calls the body object with the properly-formatted XML response portions. The response portions are placed in the buffer. When a defined buffer threshold is reached, the buffered response portions are sent to the client.

<u>The Claims Rejected Over Bayeh and Schloss</u>

Claim 1 recites a method of generating an XML document comprising:

- preparing ***only a portion*** of an XML document;
- sending said portion to a client; and
- ***repeating said preparing and said sending*** until an entire XML document is sent to a client.

In making out the rejection of this claim, the Office notes that Bayeh discloses processing and formatting results as XML, and sending a document as HTML. The Office then argues that "it would be obvious...to send the unconverted XML, as most web browsers were capable of reading XML." Applicant respectfully disagrees with the Office because first, this directly contradicts Bayeh's teachings and second, the capabilities of the most web browsers is generally irrelevant with respect to what Bayeh specifically teaches.

Bayeh specifically teaches a division of labor between data retrieval and data presentation. See, e.g. column 3, lines 62 through column 4, lines 22. Bayeh's systems and methods retrieve data using a first servlet 83, use XML to format the structured data, provide the XML data to a second servlet 85 which converts the XML data into a presentation format comprising HTML, which is then sent on to the client. Bayeh specifically teaches, in connection with its division of labor between data retrieval and data presentation, that the XML format is used as a formatting notation for the retrieved structure data (see e.g., column 8, lines 13-25), and that this retrieved structured data is then converted into a presentation format comprising the HTML (see, e.g., column 8, lines 29-34).

Bayeh further instructs on the advantages of isolating the function of the rendering servlet 85 from the function of the data servlet 83, starting in column 8 at line 36. For the convenience of the Office, this excerpt is reproduced in its entirety below:

> By isolating the function of the rendering servlet 85 from the function of the data servlet 83, the advantages of structured, modular programming are achieved. As discussed earlier, ***one of these advantages is simplifying the change process*** if changes are required to either the data retrieval logic (isolated in the data servlet 83), or to the data presentation formatting logic (isolated in the rendering servlet 85). ***Further, system throughput can be optimized*** by having multiple data servlets 83 and multiple rendering servlets 85: if one rendering servlet 85 is busy, the data servlet 83 does not have to wait to use the rendering services--it can simply pass the data to be rendered to a different rendering servlet 85'. ***This prevents bottlenecks in the system***, where one process is delayed by another process. ***System performance is also optimized in this model***, because decoupling the data retrieval logic from the presentation formatting logic allows each servlet to be optimized in its functionality. For example, a unique data servlet 83' might be created to retrieve data from a specific type of database 88 used by the server 82. Similarly, unique rendering servlets 85 might be created to format data according to different presentation requirements. When a servlet is written to retrieve data from a specific, known database (equivalently, to format data for a specific, known environment), the servlet code can be optimized for that use. ***Additionally, system flexibility is optimized*** because the unique servlets can function as pluggable components, which come and go as the needs of the environment change.

In the excerpt above, there are at least four advantages that Bayeh instructs are achieved by its system. These advantages are not insignificant to Bayeh, as Bayeh uses terms such as "optimized" to describe the results that it achieves. Yet, if Bayeh is modified in, as the Office contends, an obvious manner, to send "unconverted XML", doing so would completely eliminate the rendering servlet 85 (Fig. 4) which Bayeh describes as a necessary component to achieve all of its optimizations.

Accordingly, such is not an obvious modification to Bayeh because it would appear, at a minimum, to require removal of an essential, advantage-enhancing component of Bayeh's system (i.e. the rendering servlet). Doing so would, in effect, render Bayeh inoperative because it would require removal of an essential element. Thus, with respect to Bayeh's specific teaching, the capabilities of web browsers are irrelevant. Accordingly, for at least this reason, this claim is allowable.

In addition, the Office notes that Bayeh is silent on preparing only a portion of the XML [sic: HTML] and then repeating the steps. Applicant disagrees. Bayeh is not silent. Rather, Bayeh specifically teaches away from the concept of preparing only a portion of a document and sending only the portion to a client and then repeating the steps. Specifically, the Office's attention is directed to column 12, lines 13-15 which is provided just below:

> ***When the reformatted data stream is <u>complete</u>, Step 320 sends that HTML data stream back to the client's browser***, to be processed by the browser for presentation to the user.

That is, Bayeh is describing the processing that takes place after the HTML stream corresponding to the client's request has been processed. As Bayeh specifically states, only ***after*** the reformatted data stream is complete does the stream get sent to the client. As such, Bayeh teaches directly away from the subject matter of claim 1. Accordingly, for at least this additional reason, this claim is allowable.

The Office then cites to Schloss and argues that it teaches that an XML document can be split up into segments and processed. Based on Schloss's teaching, the Office argues that it would have been obvious to modify Schloss into Bayeh, "to allow their compatibility with networks that communicate in packets."

Applicant respectfully submits that the Office's position with respect to the combination of Schloss and Bayeh is misplaced and fails to make out a *prima facie* case of obviousness, for a couple of different reasons.

In order to make out a *prima facie* case of obviousness, three basic criteria must be met: first, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference teachings. As instructed by the Federal Circuit, "[w]hile a suggestion or motivation to combine two references may come from the general knowledge of those of ordinary skill in the art, there must be ***actual evidence*** of such a suggestion or motivation ***and the showing must be clear and particular***." *In re Dembiczak,* 50 USPQ2d 1614, 1617 (1999). Second, there must be a reasonable expectation of success. Finally, the prior art references must teach or suggest all of the claim limitations.

First, the attempted combination fails to appreciate that nothing in Bayeh, as pointed out above, supports the notion of sending only XML data to a client.

Doing so would require removal of an essential component of Bayeh's system which is, at a minimum, contrary to its teaching and, more realistically, would render it inoperative. Thus, there is no reasonable expectation that the combination suggested by the Office would succeed.

Second, and perhaps more importantly, with respect to the motivation stated by the Office to support its combination, Applicant is unaware of where this motivation comes from—does it come from the prior art, the general knowledge of those of ordinary skill in the art, or from somewhere else? As the Office surely appreciates, the motivation to combine references must come from the cited references themselves or the prior art in general. Additionally, the Federal Circuit has instructed that there must be actual evidence of such a suggestion and that the showing must be clear and particular. Applicant submits that the Office's statement of motivation ("...to allow their compatibility with networks that communicate in packets"), constitutes no such evidence, and is lacking in clarity and particularity that the Federal Circuit states that it must have.

Third, Schloss's system simply parses and analyzes object descriptions that appear in XML, identifies object fragments that meet certain criteria, revises the object description by replacing each identified object fragment with a newly created persistent identity, and sends the revised object description to a requesting node (see Figs. 3 and 4 as an example). The object fragments that meet the certain criteria and which are replaced with revised object descriptions appear to be the segments to which the Office refers. Schloss does not appear to disclose or suggest preparing ***only a portion*** of an XML document, sending the portion to a client, and ***repeating the preparing and the sending*** until an entire XML document is sent to a client. That is, Schloss appears to disclose sending an entire

XML description—whether that XML description is one that contains a complex object fragment description, or one that contains a pared down object fragment description.

Accordingly, for at least this additional reason, this claim is allowable.

Claims 2-4 depend either directly or indirectly from claim 1 and are allowable as depending from an allowable base claim. These claims are also allowable for their own recited features which, in combination with those recited in claim 1, are neither shown nor suggested in the references of record, either singly or in combination with one another.

Claim 5 recites a method of responding to an Extensible Markup Language (XML) request comprising:

- receiving an XML request from a client;
- preparing only a portion of a response to the request; and
- sending the response portion to the client.

In making out this rejection, the Office argues that this claim is rejected similarly as claim 1. Applicant submits that the subject matter of this claim is different from the subject matter of claim 1 and, as such, the Office's rationale as such pertains to claim 1 may not be appropriate in the context of claim 5. Nonetheless, Applicant respectfully points out that neither Bayeh nor Schloss appear to disclose or suggest receiving an XML request from a client, preparing only a portion of a response to the request, and sending the response portion to the client. For example, both Bayeh and Scholl appear to disclose sending complete responses relative to any requests that they receive.

Claims 6-11 and 13 depend either directly or indirectly from claim 5 and are allowable as depending from an allowable base claim. These claims are also allowable for their own recited features which, in combination with those recited in claim 5, are neither shown nor suggested in the references of record, either singly or in combination with one another. Additionally, **claim 12** is further rejected over Kavner. Given Bayeh's shortcomings, this reference adds nothing of significance to claim 12's rejection.

Claim 14 recites a method of responding to an Extensible Markup Language (XML) request comprising:

- receiving an XML request from a client;
- gathering data that is to appear in a response to the client's request;
- calling an emitter object and passing the emitter object the gathered data;
- formatting the gathered data into an appropriate XML syntax with the emitter object; and
- emitting formatted data from the emitter object, the emitter object emitting the formatted data in a manner in which an XML response can be sent to the client without having to build a hierarchical tree that represents the XML response.

In making out the rejection, the Office argues that Bayeh discloses receiving a request and cites to column 10, lines 19-25 in support therefore. While Bayeh does disclose receiving a client request, it does not disclose receiving an XML request from the client. The Office also argues that Bayeh discloses gathering data, formatting the data into XML, and sending a response. Applicant notes that Bayeh does not disclose or suggest emitting an XML response to a client. Rather, Bayeh sends an HTML response to the client.

The Office then argues that Bayeh is silent as to (1) having separate objects perform separate tasks, and (2) preparing only a portion of the XML and repeating the steps, and then relies on Schloss (as teaching that an XML document can be split up into segments and processed) to conclude that the combination of Bayeh and Schloss would render claim 14 obvious. The Office's rejection of this claim under this rationale is *non sequitur,* as claim 14 recites no such subject matter. Accordingly, for at least this reason, this claim is allowable.

In addition, this claim recites features which have not even been addressed by the Office (i.e. "the emitter object emitting the formatted data in a manner in which an XML response can be sent to the client without having to build a hierarchical tree that represents the XML response"). Accordingly, for at least this additional reason, this claim is allowable.

Claims 15-19 depend directly from claim 14 and are allowable as depending from an allowable base claim. These claims are also allowable for their own recited features which, in combination with those recited in claim 14, are neither shown nor suggested in the references of record, either singly or in combination with one another. In view of the allowability of these claims, the reference to Kavner is not seen to add anything of significance to the rejection of **claim 15**.

Claim 20 recites a method of responding to an Extensible Markup Language (XML) request comprising:

- receiving an XML request from a client, the XML request containing a Web Distributed Authoring and Versioning (WebDAV) request method;
  determining the WebDAV request method that is contained in the client's request;

creating a request method object for the WebDAV request method; gathering data that is to appear in a response to the client's request with the request method object;
- calling an emitter object and passing the emitter object data that was gathered by the request method object; and
- generating at least a portion of a syntactically correct XML response with the emitter object using the data that was gathered by the request method object.

In making out the rejection of this claim, the Office argues that Bayeh discloses (1) receiving an XML request, and (2) a servlet, which is equivalent to an object for gathering a response to the request. The Office then states that Bayeh and Schloss are silent as to passing the emitter object the data gathered and then argues that Bayeh does teach that the data servlet does the gathering itself. The Office then argues that it would be obvious to split the data servlet into two separate objects, and pass data between them. The Office's position with respect to this combination is confusing to Applicant. To the extent that this combination is relevant to the recited subject matter, Applicant submits that the Office has not stated any motivation whatsoever supporting its combination. Thus, this combination must fail for lack of any stated motivation.

The Office then goes on to state that Bayeh is silent as to generating a portion of the response, and argues that Schloss teaches that an XML document can be split up into segments and processed. Based on this, the Office argues that it would be obvious to modify Schloss into Bayeh, as generating parts of documents based on segments of incoming data was a common practice of streaming data. Applicant does not understand how this combination would render the claimed subject matter obvious. The Office then relies on Whitehead as disclosing WebDAV request methods and then, based on this disclosure, argues that a WebDav method would have inherently been determined before any further processing. Based on this, the Office argues that it would be obvious to combine Whitehead, Bayeh, and Schloss because WebDAV responses are XML.

The Office's rejection fails to make out a *prima facie* case of obviousness for any number of good reasons. First, the Office has not considered all of the subject matter of this claim. For example, this claim recites, *inter alia,* ***determining the WebDAV request method*** that is contained in the client's request, and ***creating a request method object for the WebDAV request method***. These features are entirely lacking from the cited references. Second, of those combinations made by the Office that ***do*** include a stated motivation, many if not all of the Office's stated motivations are lacking in clarity and particularity. For example, Applicant respectfully asks the Office what is clear and particular about "because WebDAV responses are XML", as the stated motivation to combine Whitehead, Bayeh, and Schloss?

Accordingly, for all of these reasons, this claim is allowable.

Claims 21-30 depend directly or indirectly from claim 20 and are allowable as depending from an allowable base claim. These claims are also allowable for their own recited features which, in combination with those recited in claim 20, are neither shown nor suggested in the references of record, either singly or in combination with one another. In view of the allowability of these claims, the reference to Kavner is not seen to add anything of significance to the rejection of **claims 27-29**.

Claim 31 recites an Extensible Markup Language (XML) request processor comprising:

an XML response generator comprising:

- a request-receiving mechanism configured to receive an XML request from a client;
- a response-preparing mechanism coupled with the request-receiving mechanism and configured to prepare ***only a portion of a response at a time***; and
- a sending mechanism coupled with the response-preparing mechanism and configured ***to receive response portions from the response-preparing mechanism and to send the response portions to the client, the sent response portions constituting less than an entirety of a response.***

In making out the rejection of this claim, the Office argues that Bayeh discloses a combined request-receiving mechanism and a response-preparing mechanism. The Office then notes that Bayeh is silent on preparing only a portion of an XML response and then repeating the steps. This is completely incorrect. Bayeh specifically teaches, with respect to its XML data stream, that the XML data stream is in its ***entirety*** before it is forwarded on for processing in its system. As an example, consider column 11, lines 20-22, which appears directly below:

> ***When the entire XML data stream required for the database results has been formatted by the data servlet***, that data stream is sent on to the next servlet in the chain at Step 270.

The Office then argues that Schloss teaches that an XML document can be split up into segments and processed. Based on this, the Office argues that it would be obvious to modify Schloss into Bayeh to allow their compatibility with networks that communicate in packets.

The Office has failed to make out a *prima facie* case of obviousness for a number of different reasons. First, the Office has misconstrued the cited references. Specifically, Bayeh teaches directly away from the properties that the

Office contends it is silent on. That is, Bayeh teaches that its XML data stream is in its entirety before it is forwarded on for further processing. How can this teaching be construed as being silent on preparing only a portion of an XML data stream? Second, the Office's motivation for making this combination is not stated with particularity and clarity. Accordingly, for at least these reasons, this claim is allowable.

Claims 32-36 depend directly from claim 31 and are allowable as depending from an allowable base claim. These claims are also allowable for their own recited features which, in combination with those recited in claim 31, are neither shown nor suggested in the references of record, either singly or in combination with one another. In view of the allowability of claim 31, the reference to Kavner is not seen to add anything of significance to the rejection of claim 36.

Claim 37 recites an Extensible Markup Language (XML) request processor comprising:

- a data-gathering object for gathering data that is to appear in a client response and generating calls in a predefined order that contain the gathered data; and
- an emitter object configured to receive calls that are generated by the data-gathering object and format the data contained therein into an appropriate XML syntax.

In making out the rejection of this claim, the Office argues that Bayeh discloses gathering data and formatting the data into XML with a data servlet. The Office further argues that Bayeh teaches the data will be formatted as an XML stream and that as such, this inherently implies a predefined order to the data.

Applicant respectfully points out that the claim at issue recites, *inter alia,* "generating ***calls*** in a predefined order". And while XML supports structured data that has a predefined order (Applicant assumes that the Office refers to the ordered nature of the XML tags that defines the structure of the data), such is not to be confused with, or assumed to be the same as the recited subject matter which recites "generating calls in a predefined order." As Bayeh discloses nothing of the sort, as such is interpreted in light of Applicant's specification, the claim is allowable for at least this reason. The Office then notes that Bayeh is silent on "passing the emitter object" the gathered data and argues the Bayeh's data servlet does the data gathering. Based on this, the Office argues that it would be obvious to split the data servlet into two separate objects and pass data between them in order to divide the work between objects.

Applicant submits that the stated motivation for making the Office's suggested modification of Bayeh is unsupported with the clarity and particularity that it must possess. Accordingly, for at least this additional reason, this claim is allowable.

Claims 38-40 depend either directly or indirectly from claim 37 and are allowable as depending from an allowable base claim. These claims are also allowable for their own recited features which, in combination with those recited in claim 37, are neither shown nor suggested in the references of record, either singly or in combination with one another. In view of the allowability of these claims, the references to Schloss and Kavner are not seen to add anything of significance to those claims rejected in view of these references.

Claim 41 recites a computer-readable medium having a computer program for responding to an XML request, the program comprising the following steps:

receiving a client request;
determining an HTTP verb that is contained in the client request;
instantiating a request method object that *corresponds to* the HTTP verb that is contained in the client request;

- using the request method object to gather information that is to appear in a response to the client's request;
- making a series of calls to an emitter object that is configured to receive information from the request method object and process the information into a *response portion* having an appropriate XML syntactic format; and
- *sending the response portion* to the client.

The Office argues that Bayeh discloses receiving a request and a servlet that is equivalent to an object for gathering a response to the request. The Office also argues that Bayeh discloses formatting the data into XML with its servlet. The Office further argues that Bayeh teaches a response sending mechanism that converts XML into HTML instead of just sending the XML. The Office then argues that it would be obvious to send the unconverted XML, as most web browsers were capable of reading XML.

That web browsers are capable of reading XML is irrelevant when one considers the specific teaching of Bayeh that instructs that the XML is converted into HTML via an architecture that has one type of servlet performing the data gathering function, and another type of data servlet perform the data formatting function. Sending only the unconverted XML would necessarily require removing a portion of Bayeh's architecture that is essential to converting from its structured data into a presentation format. Thus, for at least this reason, this claim is allowable.

The Office further argues that Bayeh is silent as to calling and emitting multiple times. Applicant notes that the claim recites, *inter alia,* "making a series of calls to an emitter object that is configured to receive information from the request method object and process the information into a response portion having an appropriate XML syntactic format, and sending the response portion to the client." The Office then argues that it would be obvious to modify Schloss into Bayeh to allow compatibility with networks that communicate in packets. The faultiness of this line of reasoning has been pointed out above.

The Office further argues that it would be obvious to call the emitter multiple times and emit multiple times in order to process these segments. This reasoning is faulty for a couple of reasons. First, the claim at issue does not recite the subject matter that the Office says it does. For example, the claim does not recite "emitting multiple times". Second, the Office gives no reason or motivation whatsoever as to why calling the emitter multiple times and emitting multiple times [sic] would be obvious.

The Office then notes that Bayeh and Schloss are silent as to a request containing an HTTP verb and notes that Whitehead discloses WebDAV request methods "which inherently would have been determined before any further processing." The Office then argues that it would have been obvious to combine Whitehead with Bayeh and Schloss "because WebDAV responses are XML".

Applicant submits that the Office has not even considered all of the subject matter recited in this claim. For example, the claim recites, *inter alia,* "determining an HTTP verb that is contained in the client request; ***instantiating a request method object that corresponds to*** the HTTP verb that is contained in the client request; using the request method object to gather information that is to appear in a response to the client's request...." The cited references fail to disclose or suggest these features. Accordingly, for at least this reason, this claim is allowable. Additionally, even if the cited references disclosed all of the features of this claim, the Office stated motivation for combining Whitehead, Bayeh and Schloss is so lacking in clarity and particularity as to provide no motivation whatsoever. Accordingly, for at least this additional reason, this claim is allowable.

Claims 42 and 43 depend directly from claim 41 and are allowable as depending from an allowable base claim. These claims are also allowable for their own recited features which, in combination with those recited in claim 41, are neither shown nor suggested in the references of record, either singly or in combination with one another. In view of the allowability of these claims, the reference to Kavner is not seen to add anything of significance to the rejection of claim 43.

Claim 44 recites a computer-readable medium having software code that is configured to receive an XML request from a client and ***instantiate an object that corresponds to an HTTP verb*** that is contained in the request. The software code further uses the object to build a portion of an XML response to the request.

This claim is rejected by the Office over the combination of Bayeh, Schloss and Whitehead. Applicant submits that the Office's stated motivation to combine these references is inadequate and, as such, the Office has failed to make out a case of *prima facie* obviousness. Accordingly, this claim is allowable.

Claims 45-47 depend directly from claim 44 and are allowable as depending from an allowable base claim. These claims are also allowable for their own recited features which, in combination with those recited in claim 44, are neither shown nor suggested in the references of record, either singly or in combination with one another.

Applicant's Response to the Office's "Response to Arguments"

The Office responded to Applicant's previous remarks in the present Office Action in the section entitled "Response to Arguments", appearing on page 12. Applicant finds it necessary to respond to each of the points raised by the Office.

With respect to the point raised by the Office in the first paragraph under Paragraph 10, Applicant has pointed out above that the capabilities of web browsers insofar as their ability to read XML is irrelevant when taken in light of Bayeh's specific teaching.

With respect to the point raised by the Office in the second paragraph under Paragraph 10, the replacement of HTML with XML would not be an obvious modification in Bayeh because the Office has failed to state a motivation to make such as modification with clarity and particularity. The fact that web browsers can read both HTML and XML does not rise to the level of a specific motivation that is stated with clarity and particularity. Assuming, *arguendo*, that it would be obvious to replace Bayeh's HTML with XML and then send the result to the client, it is not apparent how Bayeh's data presentation function would be implemented as XML is directed to how data is structured and not how data is presented.

With respect to the point raised by the Office in the third and fourth paragraphs under Paragraph 10, while Schloss may indeed process portions of the XML for the purpose of providing modified fragment descriptions, it appears that the modified XML description is sent to the client in its entirety. That is, there is no mention in Schloss of sending only portions of the modified XML description to the client.

With respect to the point raised by the Office in the fifth paragraph under Paragraph 10, Applicant submits that the Office has not addressed the subject matter of claim 14 that was added by amendment in the last response.

With respect to the point raised by the Office in the sixth paragraph under Paragraph 10, Schloss does not appear to add anything of significance as pointed out in the specific instances in which it has been discussed above.

Conclusion

All of the claims are in condition for allowance and Applicant respectfully requests a Notice of Allowability be issued forthwith. In the event that the Office's next action is anything other than issuance of a Notice of Allowability, Applicant respectfully requests that the undersigned be contacted to discuss placing the application in condition for Appeal.

Version of the Specification with Markups to Show Changes

Please replace the text in the specification starting on page 2, line 12 through page 3, line 12 with the following:

--Typically, XML documents are exchanged between different entities, such as client and server computers, in the form of requests and responses. A client might generate a request for information or a request for a certain server action, and a server might generate a response to the client that contains the information or confirms whether the certain action has been performed. In many cases, it is convenient to represent these XML documents in memory as a hierarchical tree structure. Once the hierarchical tree structure is built, the actual XML document in proper syntactic form can then be assembled. Consider the following exemplary XML code (which as had browser-executable code modified in accordance with Patent Office requirements):

```
<orders xmlns:person="http[:]_//www[.]_schemas.org/people"
        xmlns:dsig="http[:]_//dsig.org">
  <order>
    <sold-to>
      <person:name>
        <person:last-name>Layman</person:last-name>
        <person:first-name>Andrew</person:first-name>
      </person:name>
    </sold-to>
    <sold-on>1997-03-17</sold-on>
    <dsig:digital-signature>1234567890</dsig:digital-
signature>
  </order>
</orders>
```

This code includes two XML namespace declarations that are each designated with “xmlns”. The declarations include a prefix, e.g. “person” and “dsig” respectively, and the expanded namespace to which each prefix refers, e.g. “http[:]_//www[.]_schemas.org/people”, and “http[:]_//dsig.org” respectively. This code tells any reader that if an element name begins with "dsig:" its meaning is defined by whoever owns the "http[:]_//www[.]_dsig.org" namespace. Similarly, elements beginning with the "person:" prefix have meanings defined by the "http[:]_//www[.]_schemas.org/people" namespace.--

Respectfully Submitted,

Dated: 6/2/03

By: [illegible]
Lance R. Sadler
Reg. No. 38,605
(509) 324-9256